|  |  |
| --- | --- |
|  | **UNIVERSIDADE DO OESTE PAULISTA**  **CIÊNCIA DA COMPUTAÇÃO** |

**GABRYEL HENRIQUE BORGES**

**LEON BRUCHMANN RONCHI**

**INTELIGÊNCIA ARTIFICIAL I**

Presidente Prudente - SP

2024

**Relatório de Desenvolvimento do Puzzle 8**

**Objetivo**

O objetivo deste projeto é implementar uma solução para o Puzzle 8 utilizando dois algoritmos de busca: A\* e Best-First. O aplicativo deve permitir ao usuário resolver o quebra-cabeça e visualizar o caminho para a solução, além de calcular a distância de Manhattan entre os estados como função de avaliação, podendo, também, alterar o estado final do jogo.

**Estrutura do Código**

Componentes Principais:

* **Puzzle8**: Componente que renderiza o quebra-cabeça e lida com a interação do usuário ao clicar nas peças.
* A\*: Implementação da busca A\*, que considera tanto o custo atual (g) quanto uma heurística (h) para determinar o próximo estado a ser explorado.
* **Best-First**: Uma implementação de busca que utiliza apenas a heurística (h) para priorizar os estados a serem explorados.
* **Caminho**: Componente que renderiza a sequência de estados do quebra-cabeça que foi encontrada na aplicação de um dos algoritmos para que ele seja solucionado.

**Algoritmos de Busca**

**1. Algoritmo A\***

* **Descrição**: O A\* combina os custos do caminho atual (g) com uma heurística (h) para determinar o custo total (f(n) = g(n) + h(n)). A heurística utilizada é a distância de Manhattan, que calcula a soma das distâncias horizontais e verticais que cada bloco precisa se mover para alcançar sua posição final.

**2. Best-First**

* **Descrição**: Este algoritmo utiliza apenas a heurística para decidir a ordem de exploração dos estados. Embora seja menos eficiente que o A\*, é mais simples de implementar e pode ser útil para comparação.

**Comparação dos Tempos de Busca**

|  |  |  |  |
| --- | --- | --- | --- |
| A\* | | | |
| Distância Manhattan | Tempo Gasto (ms) | Nós Visitados | Caminho para a solução (qtde movimentos) |
| 20 | 1368 | 2613 | 26 |
| 15 | 1964 | 3184 | 25 |
| 10 | 82 | 689 | 20 |
| 5 | 31 | 425 | 17 |

|  |  |  |  |
| --- | --- | --- | --- |
| Best-First | | | |
| Distância Manhattan | Tempo Gasto (ms) | Nós Visitados | Caminho para a solução (qtde movimentos) |
| 20 | 9 | 177 | 36 |
| 15 | 8 | 165 | 29 |
| 10 | 8 | 160 | 24 |
| 5 | 2 | 69 | 17 |

Durante o desenvolvimento, foram realizadas várias execuções para medir o tempo de execução de ambos os algoritmos em diferentes configurações do Puzzle 8. Os tempos foram registrados em milissegundos (ms).

**Resultados dos Testes**

* Distancia Manhattan = 20 com algoritmo A\*

**Uma imagem contendo Diagrama

Descrição gerada automaticamente**

* Distancia Manhattan = 20 com algoritmo Best-First

Uma imagem contendo Texto

Descrição gerada automaticamente

* Distância Manhattan = 15 com algoritmo A\*

Texto

Descrição gerada automaticamente com confiança média

* Distância Manhattan = 15 com algoritmo Best-First

Texto

Descrição gerada automaticamente com confiança média

* Distância Manhattan = 10 com algoritmo A\*

Texto

Descrição gerada automaticamente

* Distância Manhattan = 10 com algoritmo Best-First

Texto

Descrição gerada automaticamente

* Distância Manhattan = 5 com algoritmo A\*

Texto

Descrição gerada automaticamente

* Distância Manhattan = 5 com algoritmo Best-First

Texto

Descrição gerada automaticamente

Em geral, o algoritmo A\* apresentou um melhor resultado, obtendo soluções com menos movimentos necessários para resolver o quebra-cabeça, porém ele acabou visitando muito nós e seu tempo foi maior que o apresentado pelo outro algoritmo. O Best-first, por outro lado, apresentou soluções que necessitavam de mais passos quando comparado ao A\*, porém, seu tempo de execução foi menor e os nós visitados foram menores também.

**Conclusão**

O desenvolvimento do Puzzle 8 demonstrou como diferentes algoritmos de busca podem impactar a eficiência na resolução de quebra-cabeças. O A\* provou ser o método mais eficaz para encontrar soluções com menos passos necessários, enquanto o Best-First Search, embora simples, foi útil como um ponto de comparação.

As medições de tempo e a quantidade de nós visitados foram essenciais para entender o desempenho dos algoritmos.